

INFINITE DIMENSIONAL WORD EMBEDDINGS

Eric T. Nalisnick *

Department of Computer Science
University of California, Irvine
Irvine, CA 92697, USA
enalisni@uci.edu

Sachin Ravi *

Department of Computer Science
Princeton University
Princeton, NJ 08540, USA
sachinr@cs.princeton.edu

ABSTRACT

We describe a method for learning word embeddings with stochastic dimensionality. Our *Infinite Skip-Gram* (iSG) model specifies an energy-based joint distribution over a word vector, a context vector, and their dimensionality. By employing the same techniques used to make the Infinite Restricted Boltzmann Machine (Côté & Larochelle, 2015) tractable, we define vector dimensionality over a countably infinite domain, allowing vectors to grow as needed during training. After training, we find that the distribution over embedding dimensionality for a given word is highly interpretable and leads to an elegant probabilistic mechanism for word sense induction. We show qualitatively and quantitatively that the iSG produces parameter-efficient representations that are robust to language’s inherent ambiguity.

1 INTRODUCTION

Neural language modeling (NLM) (Bengio et al., 2003; Mnih & Hinton, 2009; Turian et al., 2010; Mikolov et al., 2013) has received wide-spread attention for its ability to capture surprisingly detailed semantic information without supervision. NLM is built off the key idea that the *distributional hypothesis* (Harris, 1954) can be formulated as a prediction task: for a given word, predict its neighboring words (or vice versa). After training a classifier in this way, a *distributed representation* for each word token can be found in the model parameters. These high-dimensional, real-valued vectors—called (*Neural*) *Word Embeddings* (WEs)—have been shown to be experimentally superior to discrete representations (Baroni et al., 2014) (ostensibly) due to their ability to capture semantic subtlety and withstand ambiguity.

However, despite their success, WEs still have deficiencies. One flaw is that the vectors, since their dimensionality is fixed across the vocabulary, do not accurately reflect each word’s semantic complexity. For instance, the meaning of the word *race* varies with context (ex: car race vs biological race), but the meaning of *regatta* is rather specific and invariant. It seems unlikely that *race* and *regatta*’s representations could contain the same number of parameters without one overfitting or underfitting.

To better capture the semantic variability of words, we propose a novel embedding method that produces vectors with stochastic dimensionality. By employing the same mathematical tools that allow the definition of an *Infinite Restricted Boltzmann Machine* (Côté & Larochelle, 2015), we describe a log-bilinear energy-based model—called *Infinite Skip-Gram* (iSG)—that defines a joint distribution over a word vector, a context vector, and their dimensionality, which has a countably infinite domain. During training, the iSG allows word representations to grow naturally based on how well they can predict their context. This behavior enables the vectors of specific words to use few dimensions and the vectors of vague words to elongate as needed. Manual and experimental analysis reveals this dynamic representation elegantly captures specificity, *polysemy*, and *homonymy* without explicit definition of such concepts within the model. As far as we are aware, this is the first word em-

* Authors contributed equally.

bedding method that allows representation dimensionality to be variable and exhibit data-dependent growth¹.

2 ORIGINAL SKIP-GRAM

We first review the original *Skip-Gram* architecture before describing our novel, infinite-dimensional extension in Section 3. *Skip-Gram* (SG) (along with its sister model *Continuous Bag-of-Words* (CBOW)) is arguably the most popular method for learning word embeddings (Mikolov et al., 2013)². Its widespread use is probably due to its simplicity. SG is a computationally lean log-bilinear model whereas most previous neural language modeling had been done with neural networks.

SG learns a word’s embedding via maximizing the log probability of that word’s context (i.e. the words occurring within a fixed-sized window around the input word). Formally, let $\mathbf{w}_i \in \mathbb{R}^d$ be a d -dimensional, real-valued vector representing the i th input word w_i , and let $\mathbf{c}_k \in \mathbb{R}^d$ be a vector representing the k th context word c_k appearing in a $2K$ -sized window around an instance of w_i in some training corpus \mathcal{D} . Training the SG model reduces to minimizing the following objective function:

$$\begin{aligned}\mathcal{L}_{SG} &= \sum_{i=1}^{|\mathcal{D}|} \sum_{i-K \leq k \leq i+K, k \neq i} -\log p(c_k | w_i) \\ &= \sum_{i=1}^{|\mathcal{D}|} \sum_{i-K \leq k \leq i+K, k \neq i} -\log \frac{e^{\mathbf{c}_k^T \mathbf{w}_i}}{\sum_{v=1}^{V_c} e^{\mathbf{c}_v^T \mathbf{w}_i}} \\ &= \sum_{i=1}^{|\mathcal{D}|} \sum_{i-K \leq k \leq i+K, k \neq i} -\mathbf{c}_k^T \mathbf{w}_i + \log \sum_{v=1}^{V_c} e^{\mathbf{c}_v^T \mathbf{w}_i}\end{aligned}\tag{1}$$

where V_c is the size of the context vocabulary. Stochastic gradient descent is used to update not only \mathbf{w}_i but also \mathbf{c}_k and \mathbf{c}_v . From here forward we drop the summations over the corpus and context window to simplify notation.

Notice the cost of these gradient computations is proportional to V_c , which is usually several million or more. A *hierarchical softmax* or *negative sampling* is commonly used to alleviate the computation burden of full normalization (Mikolov et al., 2013). We’ll only consider the latter, but all models discussed can be adapted straightforwardly to use a hierarchical softmax. Negative sampling consists of randomly sampling (either from the uniform or empirical distribution) words to serve as negative examples—words that *do not* appear in the current context window—and then optimizing so that the true context word has a higher likelihood than the samples. Using negative sampling for computing Equation 1 results in the following modification:

$$-\mathbf{c}_k^T \mathbf{w}_i + \log \sum_{v=1}^{V_c} e^{\mathbf{c}_v^T \mathbf{w}_i} \approx -\mathbf{c}_k^T \mathbf{w}_i + \log \left[\sum_{s=1}^S e^{\mathbf{c}_s^T \mathbf{w}_i} + e^{\mathbf{c}_k^T \mathbf{w}_i} \right]\tag{2}$$

for S negative samples. Notice that we have defined a slightly different negative sampling SG objective than proposed in Mikolov et al. (2013), which incorporates logistic regression to discriminate the true context word from the samples. However, other embedding models such as Huang et al. (2013) have used the negative samples directly in the normalizing sum, as we do.

3 INFINITE SKIP-GRAM

We now propose our novel modification to the Skip-Gram architecture, which we call the *Infinite Skip-Gram* (iSG) model. Let word vectors $\mathbf{w}_i \in \mathbb{R}^\infty$ and context vectors $\mathbf{c}_k \in \mathbb{R}^\infty$ be infinite

¹Bartunov et al. (2015) proposes an infinite dimensional model based on Bayesian Nonparametrics (which will be reviewed in Section 4), but their model specifies an infinite number of independent, fix-sized vectors per word.

²The software *Word2Vec* provides a fast, multi-threaded implementation of both SG and CBOW: <https://code.google.com/p/word2vec/>

dimensional (instead of fixed dimensionality d), and redefine the model to be the following joint Gibbs distribution over $\mathbf{w}_i, \mathbf{c}_k$, and a random positive integer $z \in \mathbb{Z}^+$ denoting the maximum index over which to compute the vector inner product:

$$p(w_i, c_k, z) = \frac{1}{Z} e^{-E(\mathbf{w}_i, \mathbf{c}_k, z)} \quad (3)$$

where $Z = \sum_{\mathbf{w}} \sum_{\mathbf{c}} \sum_z e^{-E(\mathbf{w}, \mathbf{c}, z)}$, also known as the partition function. Define the energy function as

$$E(\mathbf{w}_i, \mathbf{c}_k, z) = z \log a - \sum_{j=1}^z w_{i,j} c_{k,j} - \lambda w_{i,j}^2 - \lambda c_{k,j}^2 \quad (4)$$

where $1 < a < \infty$, $a \in \mathbb{R}$ and λ is a weight on the L2 penalty. The $\log a$ term may seem superfluous, but, as we'll see in the next section, it is necessary for defining a convergent geometric series and controls the model's growing behavior.

3.1 A FINITE PARTITION FUNCTION

At first glance the iSGM seems intractable since the partition function, containing a sum over all countably infinite values of z , would seem to be divergent and thus incomputable. However, it is not, due to two key properties first proposed by Côté & Larochelle (2015) to define a *Restricted Boltzmann Machine* with an infinite number of hidden units (iRBM). They are:

1. **Sparsity penalty:** The L2 penalty incorporated into $E(\mathbf{w}_i, \mathbf{c}_k, z)$ (i.e. the $w_{i,j}^2$ and $c_{k,j}^2$ terms) ensures the word and context vectors must have a finite two-norm. That is to say all elements of \mathbf{w} and \mathbf{c} at an index greater than or equal to some sufficiently large index l must be zero. No proper optimization method could converge to the infinite solution if all \mathbf{w} and \mathbf{c} vectors are initialized to have a finite number of non-zero elements (Côté & Larochelle, 2015).
2. **Per-dimension constant penalty:** The energy function's $z \log a$ term results in dimensions greater than l becoming a convergent geometric series. This is discussed further below and shown in detail in the appendix.

With those two properties in mind, consider the conditional distribution of z given an input and context word:

$$p(z|w, c) = \frac{e^{-E(\mathbf{w}, \mathbf{c}, z)}}{\sum_{z'=1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, z')}}. \quad (5)$$

Again, the denominator looks problematic due to the infinite sum, but notice the following (a detailed derivation is in the appendix):

$$\begin{aligned} Z_z &= \sum_{z'=1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, z')} = \sum_{z'=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z')} + \sum_{z'=l+1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, z')} \\ &= \sum_{z'=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z')} + e^{-E(\mathbf{w}, \mathbf{c}, l)} \sum_{z'=0}^{\infty} \frac{1}{a^{z'}} \\ &= \sum_{z'=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z')} + \frac{a}{a-1} e^{-E(\mathbf{w}, \mathbf{c}, l)}. \end{aligned} \quad (6)$$

As stated in other words above, the sparsity penalty allows the sum to be split as it is in step #2 into a finite term ($\sum_{z'=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z')}$) and an infinite sum ($\sum_{z'=l+1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, z')}$) at an index l such that $w_{i,j} = c_{k,j} = 0 \forall j > l$. After $e^{-E(\mathbf{w}, \mathbf{c}, l)}$ is factored out of the second term, all remaining $w_{i,j}$ and $c_{k,j}$ terms are zero. A few steps of algebra then reveal the presence of a convergent geometric series. Intuitively, we can think of the second term, $\frac{a}{a-1} e^{-E(\mathbf{w}, \mathbf{c}, l)}$, as quantifying the data's need to expand the model's capacity given w and c .

3.2 LEARNING

We now turn our attention to learning the iSGM. We look to treat z , the random index variable, as a nuisance parameter and optimize an upper-bound to the traditional Skip-Gram objective:

$$\begin{aligned}
\mathcal{L}_{SG} &= -\log p(c_k|w_i) \\
&= -\log \sum_{z=1}^{\infty} p(c_k, z|w_i) \\
&= -\log \sum_{z=1}^{\infty} p(c_k|w_i, z)p(z|w_i) \\
&\leq \sum_{z=1}^{\infty} p(z|w_i) - \log p(c_k|w_i, z) \\
&= \mathbb{E}_{z|w}[-\log p(c_k|w_i, z)] = \mathcal{L}_{iSG}.
\end{aligned} \tag{7}$$

This expectation has, somewhat surprisingly, an analytical form, which can be seen by exploiting the same $L2$ -based arguments used to show the partition function to be finite (see the appendix):

$$\mathbb{E}_{z|w}[-\log p(c_k|w_i, z)] = -\sum_{z=1}^l p(z|w_i) \log p(c_k|w_i, z) + \frac{-a}{a-1} p(z=l|w_i) \log p(c_k|w_i, l). \tag{8}$$

However, we are unable to work with this analytical form exclusively since $\nabla_{\mathbf{w}} \mathcal{L}_{iSG}$ is infinite dimensional itself. A work-around would be to manually set l , thus making the gradients for dimensions greater than l zero and hence able to be ignored. This approach is still undesirable though since we've, for all intents and purposes, regressed to defining a finite dimensional model without the growing behavior that motivates the use of the iSGM.

We'll turn to sampling $\hat{z} \sim p(z|w)$ to produce learning that is dynamic (i.e. dimensionality can grow arbitrarily) yet tractable (i.e. a finite number of gradient updates). As suggested by Xu et al. (2015), we can calculate a Monte Carlo gradient from Equation 8 like so³:

$$\frac{\partial \mathcal{L}_{iSG}}{\partial \mathbf{w}_i} = \mathbb{E}_{z|w} \left[\frac{\partial \log p(\mathbf{c}_k|\mathbf{w}_i, z)}{\partial \mathbf{w}_i} + \frac{\partial \log p(z|\mathbf{w}_i)}{\partial \mathbf{w}_i} \log p(\mathbf{c}_k|\mathbf{w}_i, z) \right] \tag{9}$$

$$\approx \frac{1}{M} \sum_{m=1}^M \frac{\partial \log p(\mathbf{c}_k|\mathbf{w}_i, \hat{z}_m)}{\partial \mathbf{w}_i} + \frac{\partial \log p(\hat{z}_m|\mathbf{w}_i)}{\partial \mathbf{w}_i} \log p(\mathbf{c}_k|\mathbf{w}_i, \hat{z}_m) \tag{10}$$

where M is the number of samples drawn. $\frac{\partial \mathcal{L}_{iSG}}{\partial \mathbf{c}_k}$ is of the same form.

Yet there's still a problem in that $\hat{z} \in [1, \infty)$ and therefore a very large dimensionality (say, a few thousand or more) could be sampled, resulting in the gradient incurring painful computational costs and variance. To remedy this situation, if a \hat{z} value greater than the current value of l is sampled, we set $\hat{z} = l + 1$, restricting the model to grow only one dimension at a time (just as done for the iRBM). Constraining growth in this way is computationally efficient since \hat{z} can be drawn from a $(l + 1)$ -dimensional multinoulli distribution with parameters

$$\Theta = \left\{ \theta_1 = p(z=1|w), \dots, \theta_l = p(z=l|w), \theta_{l+1} = P(z > l|w) = \frac{a}{a-1} p(z=l|w) \right\}.$$

The intuition is the model can sample a dimension less than or equal to l if l is already sufficiently large or draw the $(l + 1)$ th option if not, choosing to increase the model's capacity. Notice a controls this growing behavior: as a approaches one (from the right), $P(z > l|w)$ approaches infinity.

Lastly, we'll mention two relaxations that make learning less computationally burdensome. The first is that the conditional probabilities $p(c_k|w_i, \hat{z}_m)$ can be computed with negative sampling, just like in the original SG (as described in Equation 2). The second is for computing $p(z|w_i)$. It also requires

³The $\frac{\partial \log p(z|\mathbf{w}_i)}{\partial \mathbf{w}_i}$ comes from the identity $\frac{\partial p(z|\mathbf{w}_i)}{\partial \mathbf{w}_i} = p(z|\mathbf{w}_i) \frac{\partial \log p(z|\mathbf{w}_i)}{\partial \mathbf{w}_i}$. The former term on the RHS is factored out to form an expectation.

a sum over the context vocabulary, but instead of randomly sampling words, we'll marginalize only over the context words in the current window. Formally, the approximation can be written as:

$$p(z|w_i) = \frac{\sum_{v=1}^{V_c} e^{-E(\mathbf{w}, \mathbf{c}_v, z)}}{\sum_{z'=1}^{\infty} \sum_{v=1}^{V_c} e^{-E(\mathbf{w}, \mathbf{c}_v, z')}} \approx \frac{\sum_{\mathbf{c}_v \in C_i} e^{-E(\mathbf{w}, \mathbf{c}_v, z)}}{\sum_{z'=1}^{\infty} \sum_{\mathbf{c}_v \in C_i} e^{-E(\mathbf{w}, \mathbf{c}_v, z')}} \quad (11)$$

where V_c is the context vocabulary size and C_i is the multiset of context words appearing in a $2K$ -sized window around input word w_i (i.e. at indices k such that $i - K \leq k \leq i + K, k \neq i$). Using the current context for approximate marginalization is preferred over random words because it will concentrate $p(z|w_i)$ around the dimensions most appropriate for modeling the current context and as a result reduce sampling variance.

3.3 PREDICTION, SIMILARITIES, AND DISAMBIGUATION

After training, the iSG can be used for tasks such as predicting context words, computing similarity between words, and inferring a word's sense or meaning. Prediction of context word c_k given word w_i will be done with the complete-data likelihood: $p(c_k|w_i) = \sum_{z=1}^l p(c_k, z|w_i)$ where l is the max dimension expanded to during learning.

Computing similarities is a little more arbitrary. Traditionally, cosine similarity has been used to determine distance between (original) SG embeddings w_i and w_j :

$$\text{sim}_{i,j}^{\text{SG}} = \cos(\mathbf{w}_i, \mathbf{w}_j). \quad (12)$$

Notice context vectors are not used. In fact, *Word2Vec* discards the context vectors after training. Similarities between iSG vectors, on the other hand, should not be computed with cosine similarity. Vector length encodes semantics (correlating with word specificity), and angular distance destroys this information. The expected inner product is a better choice for comparing iSG vectors:

$$\text{sim}_{i,j}^{\text{iSG}} = \sum_{z=1}^l p(z|w_i, w_j) \mathbf{w}_i^T \mathbf{w}_j|_z \quad (13)$$

where $|_z$ is notational shorthand denoting that the inner product be taken over z dimensions. If only one word is specified—such as when retrieving nearest neighbors—distances should be computed with an asymmetric version of $\text{sim}_{i,j}^{\text{iSG}}$:

$$\text{sim}_{i,-}^{\text{iSG}} = \sum_{z=1}^l p(z|w_i) \mathbf{w}_i^T \mathbf{w}_j|_z \quad (14)$$

where here the probability of each dimension, $p(z|w_i)$, is conditioned on only the specified word. Also note that taking the inner product between input and context vectors ($\mathbf{w}_i^T \mathbf{c}_j$) yields qualitatively similar results.

Lastly, inferring word w_i 's senses is done through examination of its distribution over dimensionalities: $p(z|w_i) = \frac{1}{Z_{z,c}} \sum_{\mathbf{c}_v} e^{-E(\mathbf{w}, \mathbf{c}_v, z)}$ where the sum is taken over the full context vocabulary. We should expect each sense and meaning of a word to be represented by a mode in $p(z|w_i)$. Interestingly, we can isolate one of these modes by summing over a subset of the vocabulary that places probability on the same or nearby dimensions. This is essentially what is done during training by summing over the current context only when computing $p(z|w_i)$. Selective (approximate) marginalization is useful not only for computational reasons but for modulating similarities scores. For example, as we will show in Section 5, computing $p(z|w = \text{'net'})$ by summing over selected sports-related words and using the approximate distribution in $\text{sim}_{i,-}^{\text{iSG}}$ retrieves *football* as *net*'s nearest neighbor. Yet, when technology-related words are summed over, *www* is the nearest neighbor.

4 RELATED WORK

Much of the existing work has attempted to improve upon Skip-Gram's fixed-sized vector representations by adding parameters. One such effort is described in Vilnis & McCallum (2014); they represent each word with a multivariate Gaussian distribution. The Gaussian embedding's mean

parameter serves much like a traditional vector representation, anchoring the word into a semantic location in high-dimensional space. The method’s novelty and utility arises from the covariance parameter’s ability to elegantly capture word specificity (i.e. specific words have small covariances, vague words have large ones). Other notable extensions include Kiros et al. (2014), which employed tensors to incorporate meta-data into the embedding, and Pennington et al. (2014), which leveraged global context as well as local during learning.

Other previous work has proposed linguistically motivated extensions, specifically to handle *polysemy* and *homonymy*. Existing approaches of this kind can be split into two categories: one in which the number of word senses are learned a priori to vectors and one in which senses and vectors are learned jointly. Huang et al. (2012) and Reisinger & Mooney (2010) fall into the first category as they use a preprocessing step to cluster all observed context windows (of some fixed size) appearing around each occurrence of a word. Each cluster is assumed to represent a different meaning or sense, and thus a new (independent) vector representation is created for each cluster. Chen et al. (2014) also propose a two stage algorithm. Yet, their work is not appropriate for comparison since they leverage WordNet to learn senses after training a traditional Skip-Gram model.

Members of the second category include Neelakantan et al. (2015), Tian et al. (2014), and Bartunov et al. (2015). Neelakantan et al. (2015)’s *MSSG* model performs clustering to identify distinct senses (similarly to the work in the first category); however, they estimate an input word’s cluster membership from the current value of the context representations. Learning the context vectors in turn results in the clusters becoming learned and adaptive. They also describe a nonparametric variant (NP-MSSG) that creates a new cluster if an observed context is sufficiently different from existing clusters. Tian et al. (2014) describe a probabilistic model that is similar in spirit to the MSSG. Word sense is defined as a latent variable indexing a fixed number of independent vector representations for a given word. Expectation-Maximization (EM) is used for learning. Lastly, Bartunov et al. (2015)’s *AdaGram* model can be seen as an infinite dimensional extension of Tian et al. (2014); it uses the Dirichlet Process from Bayesian Nonparametrics to define an infinite number of vectors for each word.

Our iSG model sits conceptually between parameter-expanded embeddings without explicit linguistic motivations (such as Vilnis & McCallum (2014)) and embeddings with a latent index over prototypes (such as Tian et al. (2014) or Bartunov et al. (2015)). It differentiates itself from previous work in the following ways. Firstly, as mentioned in the Introduction, no previous work has made the embedding dimensionality stochastic, allowing vectors to exhibit unconstrained, data-dependent growth. The latent variable models mentioned learn *multiple independent vectors of fixed dimensionality*, enabling a costly replication of the full vocabulary embedding matrix. Secondly, all of the models mentioned *add* parameters in order to gain capacity and functionality. The iSG, on the other hand, only contains $2|V| \times z_{\max}$ parameters (where $|V|$ is the vocabulary size and z_{\max} is the max dimension learned), which is the same number as traditional SG with a vector size set to z_{\max} . In some sense, iSG owes its increased functionality to its ability to use *less* parameters, as it can truncate vectors at different dimensions to create different meanings. And lastly, as we’ll demonstrate in Section 5, iSG captures multiple word meanings without explicit definition of such concepts and requires no pre-processing and/or ad-hoc clustering (as needs to be done in Huang et al. (2012), Reisinger & Mooney (2010), and Neelakantan et al. (2015)).

5 EVALUATION

We now evaluate our Infinite Skip-Gram (iSG) model qualitatively and quantitatively against original Skip-Gram (SG). For all experiments, iSG and SG were trained on a one billion word subset of Wikipedia (6/29/08 snapshot). Three SG models with dimensionalities 100, 300, and 500 were trained with the Word2Vec implementation. Nine iSG models were trained using combinations of $a = \{1.1, 1.075, 1.05\}$ and $\lambda = \{1 \times 10^{-4}, 1 \times 10^{-6}, 1 \times 10^{-8}\}$, eaching taking approximately twelve hours on a c3.8XL Amazon EC2 instance using thirty threads. The same learning rate ($\alpha = 0.05$), decay schedule for α (Word2Vec default), number of negative samples (5), context window size (10), and number of training epochs (1) were used for SG and iSG. iSG was initialized to two dimensions for every run and produced vectors of lengths ranging from 275 to 536 dimensions. We attempted to use AdaGrad, but experiments showed it made vectors grow to extreme lengths.

	Infinite Skip-Gram (iSG)			Skip-Gram (SG)
	full vocabulary	<i>car, driver, track, speed, motor</i>	<i>politics, election, vote, washington, campaign</i>	n/a
race	households	laps	election	racers
	latino	races	elections	racing
	couples	busch	electoral	prix
	hispanic	prix	party	finish
	asian	nascar	seats	drivers
	full vocabulary	<i>goal, score, win, soccer, hockey</i>	<i>web, email, online, technology, website</i>	n/a
net	num	football	www	com
	score	footballers	http	www
	income	nfl	php	homepage
	goals	league	com	http
	liga	fc	https	html
	full vocabulary	<i>money, finance, loan, savings, investment</i>	<i>river, stream, water, shore, sand</i>	n/a
bank	investment	investment	river	banks
	banking	banking	railway	investment
	securities	securities	road	financial
	corporation	corporation	avenue	insurance
	holdings	currency	highway	finance

Table 1: The five nearest neighbors for *race*, *net*, and *bank* learned by iSG (left three columns, using Equation 14) and SG (rightmost column, using Equation 12) are displayed above. The three columns under iSG denote neighbors for different computations of $p(z|w)$. In the first, the full vocabulary was summed over for marginalization. In the second and third columns, only a select few words were summed over (shown in italic), and as a result different senses are evoked.

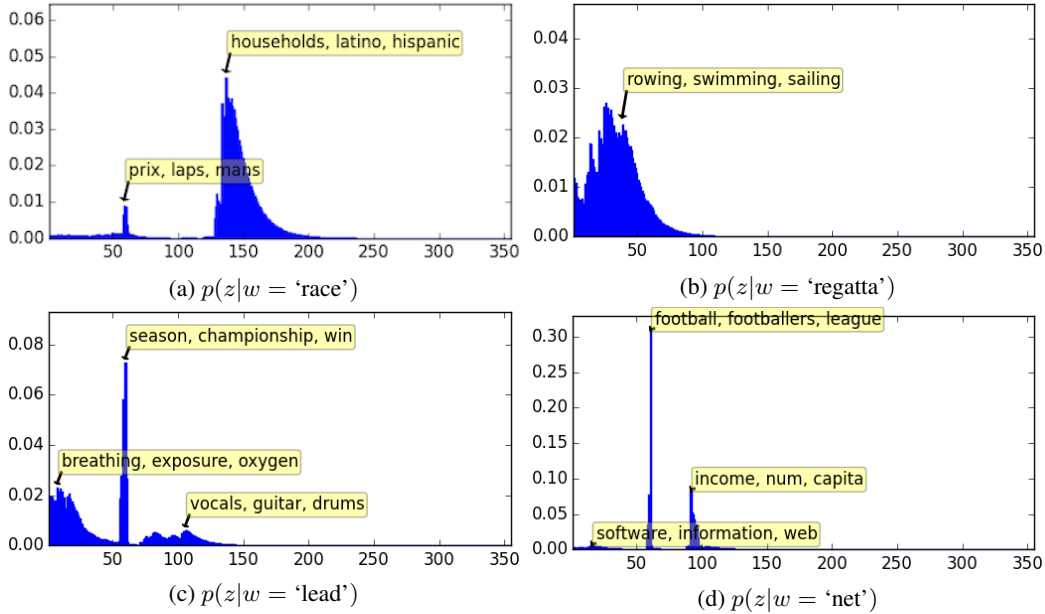


Figure 1: The four plots above show $p(z|w)$ for the words *race* (a), *regatta* (b), *lead* (c), and *net* (d). We see the homographs (a, c, d) have more complicated distributions (more modes, probability mass at larger dimensions) than the specific word *regatta*. This suggests iSG is adapting capacity as expected.

5.1 QUALITATIVE RESULTS

We begin evaluation by examining the qualitative properties of the embeddings learned by iSG. We selected four homographs—*bank*, *race*, *net*, and *lead*—to see if iSG can capture their various facets of meaning. We present nearest neighbor results for the first three and visualize $p(z|w)$ for the last three.

5.1.1 NEAREST NEIGHBORS

As has become standard practice when presenting an embedding model, we show that nearest neighbors in the learnt space are semantically related. Table 1 contains the five nearest neighbors for the words *race*, *net*, and *bank* learned by SG and iSG (calculated by Equations 12 and 14 respectively). The ten nearest neighbors for the same words can be found in Table 2 in the appendix. The leftmost column shows iSG-produced neighbors when using full marginalization to compute the expected inner product. The second and third columns show neighbors when approximate marginalization is used to surface different senses of each word. Original SG’s neighbors are in the fourth column. Comparing iSG’s first-column neighbors to SG’s, we see that they are all but indistinguishable in the case of *bank*, but neighbors are much different for *race* and *net*. iSG seems to have found terms related to the biological sense of *race* as the most globally similar while SG finds car racing terms exclusively. For *net*, SG has only learned an internet-related meaning, but iSG’s global results include a mixture of sporting (*goals*, *liga*) and financial/numerical (*income*, *score*) terms.

While perhaps original SG’s neighbors are slightly more focused than iSG’s first-column neighbors, we see quite conspicuously that SG has failed to recognize that these words are homographs. That is, SG only captures one meaning in its representation. Conversely, iSG was able to capture various meanings, which is made clear by the second and third columns of Table 1 in which the similarities were calculated by summing over the five words in italic (ex: *car*, *driver*, *track*, *speed*, *motor*) when computing $p(z|w)$. Quite dramatically different senses are able to be found through this simple modification, including the senses found by SG. Furthermore, we find that iSG’s vectors are robust to marginalizing with improper contexts. For instance, calculating *race*’s nearest neighbors by marginalizing over *investment*, *banking*, *loan*, *savings*, *investment* results in the following five nearest neighbors: *laps*, *racing*, *lap*, *ferrari*, *prix*. No banking-related terms were found that, if present, would suggest spurious semantic relationships could be induced by arbitrary choices of words.

5.1.2 MODES OF MEANING

Next we show iSG’s potential for meaning disambiguation by plotting $p(z|w)$ for the selected homographs and the word *regatta* (from our example in the Introduction); see Figure 1. The yellow boxes contain the three nearest neighbors as determined by inner product computed over the dimensionality specified by the arrow.

We see that, as expected, *race* (a) exhibits a much more complicated distribution over dimensions than *regatta* (b) does. $p(z|w = \text{‘race’})$ features at least two prominent modes corresponding to *automobile* and *biological* meanings. $p(z|w = \text{‘regatta’})$ is clearly uni-modal, concentrating around its only meaning (boat race). Moreover, *race* is a longer vector in the sense that it puts probability mass on as many as 200 dimensions. *regatta*, on the other hand, has no perceptible probability mass after 50 dimensions.

We see similarly interesting distributions from the two other homographs plotted, *lead* (c) and *net* (d). The former appears to have three strong modes corresponding to *lead poisoning*, *leader of a sports team or statistical category*, and *leader in a band* (ex: ‘lead trumpet’). The latter also has three distinct modes evoking technology, sports, and numerical meanings. The technology mode is hard to see in Figure 1, but magnifying will reveal a clear plateau. Notice too the scale of the plot is different than the others due to the dominant middle mode. And lastly, note that the sports sense common to all words—*prix* in (a), *rowing* in (b), *season* in (c), and *football* in (d)—occurs at roughly the same place in each distribution, at slightly above 50 dimensions.

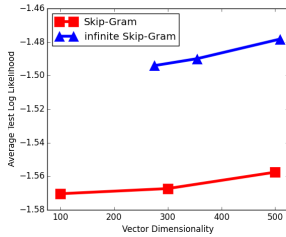


Figure 2: Context prediction. This Figure shows the average test log likelihood for iSG (blue) and SG (red) at three different vector dimensionalities. Since the vector length of iSG cannot be specified, models with vectors closest in size to 100, 300, and 500 dimensions were chosen. iSG vectors had lengths 275, 355, and 509. iSG is clearly better at predicting context words *despite having the same number of parameters as SG*.

5.2 EXPERIMENT: CONTEXT PREDICTION

We compared the ability of SG and iSG to predict the context of a given word on a held out test set, which was created from the last thirty-two million words of the same Wikipedia snapshot. In Figure 2 we plot the average log likelihood computed by SG (i.e. $\frac{1}{|(w_i, c_k)|} \sum_{(w_i, c_k)} \log p(c_k | w_i)$) with vectors of dimensionality 100, 300, 500 against the iSG average complete-data log likelihood (i.e. $\frac{1}{|(w_i, c_k)|} \sum_{(w_i, c_k)} \log \sum_z p(c_k, z | w_i)$) for 275, 355, and 509 dimensional vectors. These vectors were chosen because their sizes were most comparable to the SG vector dimensionalities. Notice that even though SG and iSG *have almost exactly the same number of parameters*, iSG has a conspicuously higher log likelihood.

6 CONCLUSIONS AND FUTURE WORK

We’ve proposed a novel word embedding model called *Infinite Skip-Gram* that defines vector dimensionality as a random variable with a countably infinite domain. Training via the Monte Carlo gradient proposed in Section 3.2 allows embeddings to grow as the data requires. This property is especially well suited for learning representations of homographs—which Skip-Gram notably fails at—and this is demonstrated in Section 5.1. A unique quality of the iSG is that it is highly interpretable (in comparison to other embedding methods) due to its ability to produce a distribution over the dimensionality of a given word ($p(z|w)$). Plots of $p(z|w)$ such as the ones in Figure 1 concisely show how specific/vague a word is and its various senses just from the mode landscape. Soon we will add more experimental results on word similarity and sense induction tasks.

REFERENCES

- Baroni, Marco, Dinu, Georgiana, and Kruszewski, Germán. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 238–247, 2014.
- Bartunov, Sergey, Kondrashkin, Dmitry, Osokin, Anton, and Vetrov, Dmitry. Breaking sticks and ambiguities with adaptive skip-gram. *arXiv preprint arXiv:1502.07257*, 2015.
- Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Janvin, Christian. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.
- Chen, Xinxiong, Liu, Zhiyuan, and Sun, Maosong. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1025–1035, 2014.
- Côté, Marc-Alexandre and Larochelle, Hugo. An infinite restricted boltzmann machine. *arXiv preprint arXiv:1502.02476*, 2015.
- Harris, Zellig S. Distributional structure. *Word*, 1954.
- Huang, Eric H., Socher, Richard, Manning, Christopher D., and Ng, Andrew Y. Improving word representations via global context and multiple word prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2012.

- Huang, Po-Sen, He, Xiaodong, Gao, Jianfeng, Deng, Li, Acero, Alex, and Heck, Larry. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 2333–2338. ACM, 2013.
- Kiros, Ryan, Zemel, Richard, and Salakhutdinov, Ruslan R. A multiplicative model for learning distributed text-based attribute representations. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2014.
- Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.
- Mnih, Andriy and Hinton, Geoffrey E. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pp. 1081–1088, 2009.
- Neelakantan, Arvind, Shankar, Jeevan, Passos, Alexandre, and McCallum, Andrew. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015.
- Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.
- Reisinger, Joseph and Mooney, Raymond J. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 109–117. Association for Computational Linguistics, 2010.
- Tian, Fei, Dai, Hanjun, Bian, Jiang, Gao, Bin, Zhang, Rui, Chen, Enhong, and Liu, Tie-Yan. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING*, pp. 151–160, 2014.
- Turian, Joseph, Ratinov, Lev, and Bengio, Yoshua. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394. Association for Computational Linguistics, 2010.
- Vilnis, Luke and McCallum, Andrew. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.
- Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Courville, Aaron, Salakhutdinov, Ruslan, Zemel, Richard, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

APPENDIX

FINITE PARTITION FUNCTION

Below we show that summing over $z \in [1, \infty)$, as needs to be done to calculate the partition function, results in a finite value by way of a convergent geometric series.

$$\begin{aligned}
Z_z &= \sum_{z=1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, z)} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + \sum_{z=l+1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, z)} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + \sum_{z=l+1}^{\infty} e^{-E(\mathbf{w}, \mathbf{c}, l) - (z-l-1) \log a + \sum_{i=l+1}^z w_i c_i - w_i^2 - c_i^2} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + e^{-E(\mathbf{w}, \mathbf{c}, l)} \sum_{z=l+1}^{\infty} e^{-(z-l-1) \log a + \sum_{i=l+1}^z w_i c_i - w_i^2 - c_i^2} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + e^{-E(\mathbf{w}, \mathbf{c}, l)} \sum_{z=l+1}^{\infty} e^{-(z-l-1) \log a} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + e^{-E(\mathbf{w}, \mathbf{c}, l)} \sum_{z=0}^{\infty} e^{-z \log a} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + e^{-E(\mathbf{w}, \mathbf{c}, l)} \sum_{z=0}^{\infty} \frac{1}{a^z} \\
&= \sum_{z=1}^l e^{-E(\mathbf{w}, \mathbf{c}, z)} + \frac{a}{a-1} e^{-E(\mathbf{w}, \mathbf{c}, l)}.
\end{aligned}$$

FINITE EXPECTATION

Next we show that the iSG objective—specifically the expectation over $p(z|w)$ needed for its calculation—is finite. Again we use the argument that every word and context vector has non-zero elements only up to and including some index l .

$$\begin{aligned}
\mathbb{E}_{z|w}[-\log p(c_k|w_i, z)] &= -\sum_{z=1}^{\infty} p(z|w_i) \log p(c_k|w_i, z) \\
&= -\sum_{z=1}^l p(z|w_i) \log p(c_k|w_i, z) + -\sum_{z=l+1}^{\infty} p(z|w_i) \log p(c_k|w_i, z).
\end{aligned}$$

Clearly the first term is finite, and thus we turn our attention exclusively to the second term:

$$\begin{aligned}
\sum_{z=l+1}^{\infty} p(z|w_i) \log p(c_k|w_i, z) &= \sum_{z=l+1}^{\infty} p(z|w_i) \log \left[\frac{e^{-E(\mathbf{w}_i, \mathbf{c}_k, z)}}{\sum_{\mathbf{c}'} e^{-E(\mathbf{w}_i, \mathbf{c}', z)}} \right] \\
&= \sum_{z=l+1}^{\infty} p(z|w_i) \log \left[\frac{e^{-E(\mathbf{w}_i, \mathbf{c}_k, l) - (z-l-1) \log a + \sum_{j=l+1}^z w_{i,j} c_{k,j} - w_{i,j}^2 - c_{k,j}^2}}{\sum_{\mathbf{c}'} e^{-E(\mathbf{w}_i, \mathbf{c}', l) - (z-l-1) \log a + \sum_{j=l+1}^z w_{i,j} c'_{j} - w_{i,j}^2 - c'^2_{j}}} \right] \\
&= \sum_{z=l+1}^{\infty} p(z|w_i) \log \left[\frac{e^{-E(\mathbf{w}_i, \mathbf{c}_k, l) - (z-l-1) \log a}}{\sum_{\mathbf{c}'} e^{-E(\mathbf{w}_i, \mathbf{c}', l) - (z-l-1) \log a}} \right] \\
&= \sum_{z=l+1}^{\infty} p(z|w_i) \log \left[\frac{e^{-(z-l-1) \log a} e^{-E(\mathbf{w}_i, \mathbf{c}_k, l)}}{e^{-(z-l-1) \log a} \sum_{\mathbf{c}'} e^{-E(\mathbf{w}_i, \mathbf{c}', l)}} \right] \\
&= \log p(c_k|w_i, l) \sum_{z=l+1}^{\infty} p(z|w_i) \\
&= \log p(c_k|w_i, l) \frac{1}{Z_{\mathbf{c}, z}} \sum_{z=l+1}^{\infty} \sum_{\mathbf{c}'} e^{-E(\mathbf{w}_i, \mathbf{c}', z)} \\
&= \log p(c_k|w_i, l) \frac{1}{Z_{\mathbf{c}, z}} \sum_{\mathbf{c}'} e^{-E(\mathbf{w}_i, \mathbf{c}', l)} \sum_{z=l+1}^{\infty} e^{-(z-l-1) \log a} \\
&= \frac{a}{a-1} p(z=l|w_i) \log p(c_k|w_i, l).
\end{aligned}$$

Combining this result with the summation over the first l terms, we see the expectation's analytical form is

$$\mathbb{E}_{z|w}[-\log p(c_k|w_i, z)] = - \sum_{z=1}^l p(z|w_i) \log p(c_k|w_i, z) + \frac{-a}{a-1} p(z=l|w_i) \log p(c_k|w_i, l).$$

Infinite Skip-Gram (iSG)			Skip-Gram (SG)
race			
full vocabulary	<i>car, driver, track, speed, motor</i>	<i>politics, election, vote, washington, campaign</i>	n/a
households	laps	election	races
latino	races	elections	racing
couples	busch	electoral	prix
hispanic	prix	party	finish
asian	nascar	seats	drivers
races	drivers	senate	driver
islander	lap	republican	pole
census	racing	parliament	standings
laps	ferrari	democratic	driving
population	asian	elected	qualifying
net			
full vocabulary	<i>goal, score, win, soccer, hockey</i>	<i>web, email, online, technology, website</i>	n/a
num	football	www	com
score	footballers	http	www
income	nfl	php	homepage
goals	league	com	http
liga	fc	https	html
avg	nhl	html	htm
midfielder	baseball	index	asp
gross	hockey	web	web
yards	basketball	org	website
nhl	rugby	software	links
bank			
full vocabulary	<i>money, finance, loan, savings, investment</i>	<i>river, stream, water, shore, sand</i>	n/a
investment	investment	river	banks
banking	banking	railway	investment
securities	securities	road	financial
corporation	corporation	avenue	insurance
holdings	currency	highway	finance
currency	dollar	situated	capital
capital	holdings	located	trading
dollar	saudi	lake	exchange
owned	capital	bridge	fund
finance	investors	street	credit

Table 2: Nearest Neighbors. The table above is an expanded version of Table 1, displaying the ten nearest neighbors for *race*, *net*, and *bank* as learned by Infinite Skip-Gram (left three columns, retrieved via Equation 14) and original Skip-Gram (rightmost column, retrieved via Equation 12).